**Department of Computer Science and Engineering**

**National Institute of Technology, Hamirpur**

**Name: Aryan Puri**

**Roll Number: 22BEC032**

**Digital Image Processing CS-325**

**Solution Laboratory Assignment -5**

**Topic: Image Segmentation using Otsu’s Method of thresholding**

**Question 1: Design a program to read an image (Scene.jpg as given) and apply Otsu’s Method of thresholding on each plane (total 3 planes). Find the optimal threshold that divide (segment) the image (single plane) into binary levels of intensity. Display the input image and processed (output) image with respective histogram. Also, print the between class variance (σB2) and optimal threshold(k\*) efficiency matric (η). Write the conclusion based on the observation of the output image.**

**Solution:**

**Python Program:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

img=cv2.imread('./Assignment 5/img.jpg',1)

rows,cols,channels=img.shape

def OstuMethod(img):

    ihist=[0]\*256

    rows,cols=img.shape

    for i in range(rows):

        for j in range(cols):

            ihist[img[i][j]]+=1

    for i in range(256):

        ihist[i]/=(rows\*cols)

    Mg=0

    for i in range(256):

        Mg+=(i\*ihist[i])

    P1=[0]\*256

    M=[0]\*256

    P1[0]=ihist[0]

    M[0]=0

    Sigma2B=[0]\*256

    maxk=0

    for k in range(1,254):

        P1[k]=P1[k-1] + ihist[k]

        M[k]=M[k-1] + ihist[k]\*k

        Sigma2B[k]= (pow(P1[k]\*Mg - M[k], 2) / (P1[k]\*(1-P1[k])))

        if(Sigma2B[maxk]<Sigma2B[k]):

            maxk=k

    return (ihist,Sigma2B,maxk)

B,G,R=cv2.split(img)

irhist,rSigma2B,rmaxk=OstuMethod(R)

ighist,gSigma2B,gmaxk=OstuMethod(G)

ibhist,bSigma2B,bmaxk=OstuMethod(B)

rmaxkgraph=[0]\*256

gmaxkgraph=[0]\*256

bmaxkgraph=[0]\*256

rmaxkgraph[rmaxk]=rSigma2B[rmaxk]

gmaxkgraph[gmaxk]=gSigma2B[gmaxk]

bmaxkgraph[bmaxk]=bSigma2B[bmaxk]

plt.subplot()

plt.title('Variation of Sigma2B with k')

plt.plot(rSigma2B,label='Red Sigma2B')

plt.plot(rmaxkgraph,label='Red Kmax')

plt.plot(gSigma2B,label='Green Sigma2B')

plt.plot(gmaxkgraph,label='Green Kmax')

plt.plot(bSigma2B,label='Blue Sigma2B')

plt.plot(bmaxkgraph,label='Blue Kmax')

plt.legend()

plt.show()

Rthresh=np.zeros((rows,cols),dtype=np.uint8)

Gthresh=np.zeros((rows,cols),dtype=np.uint8)

Bthresh=np.zeros((rows,cols),dtype=np.uint8)

orhist=[0]\*256

oghist=[0]\*256

obhist=[0]\*256

for i in range(rows):

    for j in range(cols):

        if(R[i][j]>rmaxk):

            Rthresh[i][j]=255

            orhist[255]+=1

        else:

            orhist[0]+=1

        if(G[i][j]>gmaxk):

            Gthresh[i][j]=255

            oghist[255]+=1

        else:

            oghist[0]+=1

        if(B[i][j]>bmaxk):

            Bthresh[i][j]=255

            obhist[255]+=1

        else:

            obhist[0]+=1

orhist[0]/=(rows\*cols)

orhist[255]/=(rows\*cols)

oghist[0]/=(rows\*cols)

oghist[255]/=(rows\*cols)

obhist[0]/=(rows\*cols)

obhist[255]/=(rows\*cols)

images=[R,G,B,irhist,ighist,ibhist,Rthresh,Gthresh,Bthresh,orhist,oghist,obhist]

plt.title('Input Image Red Green Blue Plane')

for i in range(3):

    plt.subplot(1,3,i+1)

    plt.imshow(images[i],'grey')

    plt.xticks([]),plt.yticks([])

plt.show()

plt.title('Input Image Red Green Blue Plane Histograms')

for i in range(3,6):

    plt.subplot(1,3,i-2)

    plt.plot(images[i])

plt.show()

plt.title('Output Thresholded Red Green Blue Plane')

for i in range(6,9):

    plt.subplot(1,3,i-5)

    plt.imshow(images[i],'grey')

    plt.xticks([]),plt.yticks([])

plt.show()

plt.title('Output Thresholded Red Green Blue Plane Histogram')

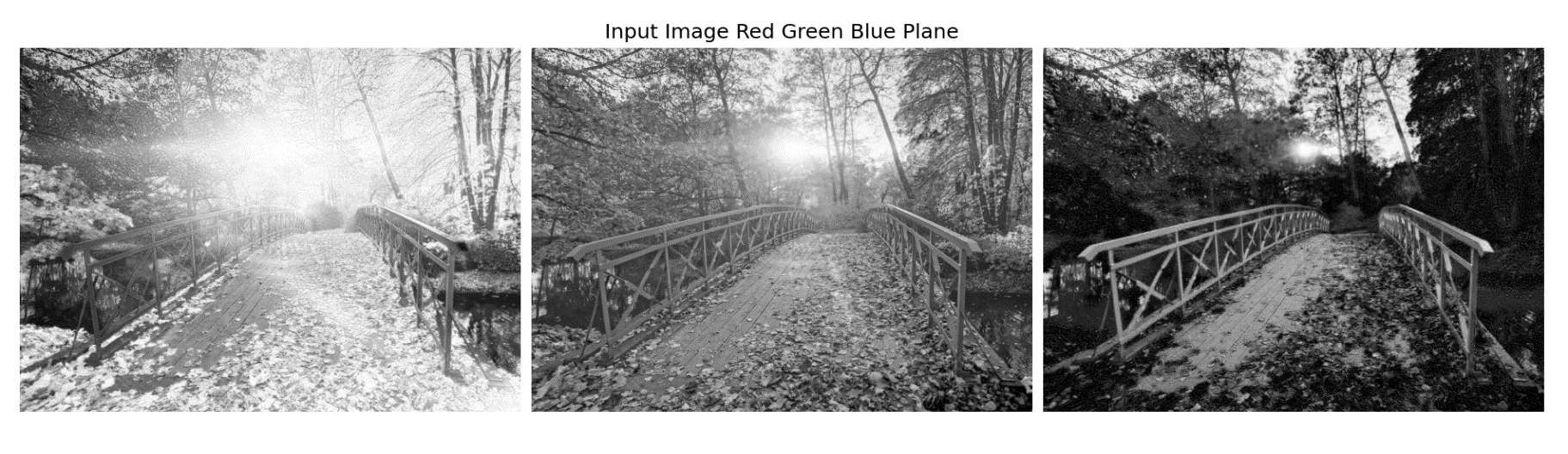
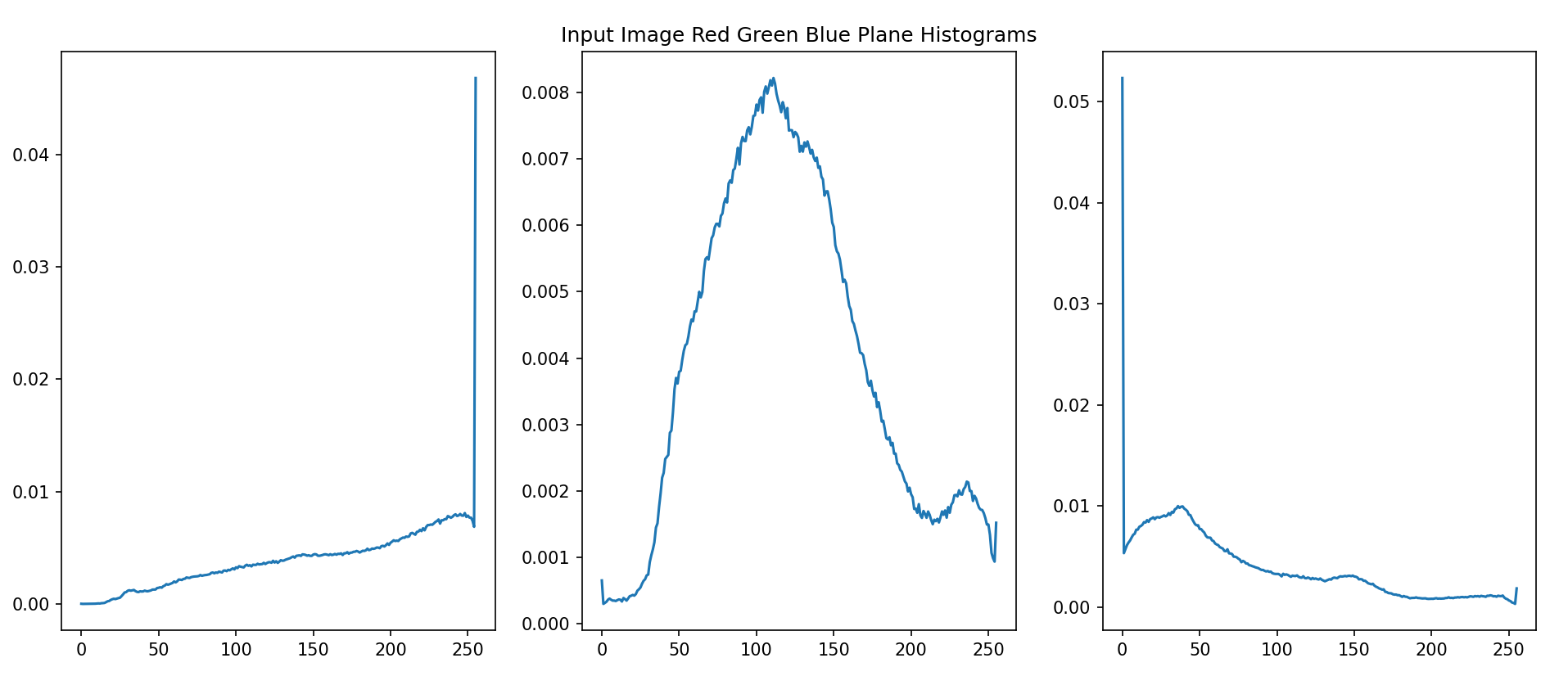
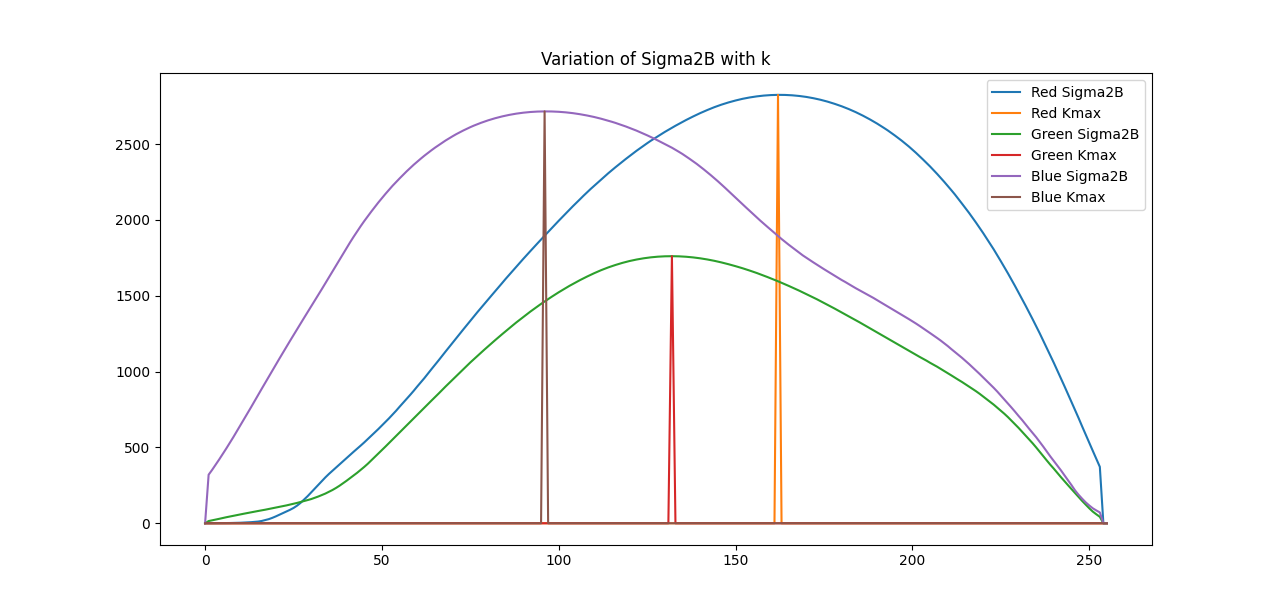
for i in range(9,12):

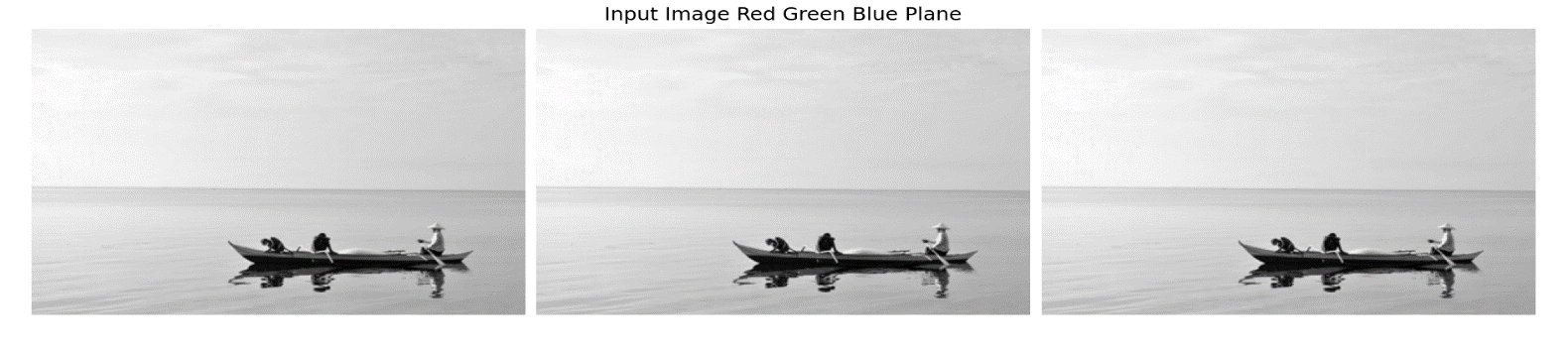
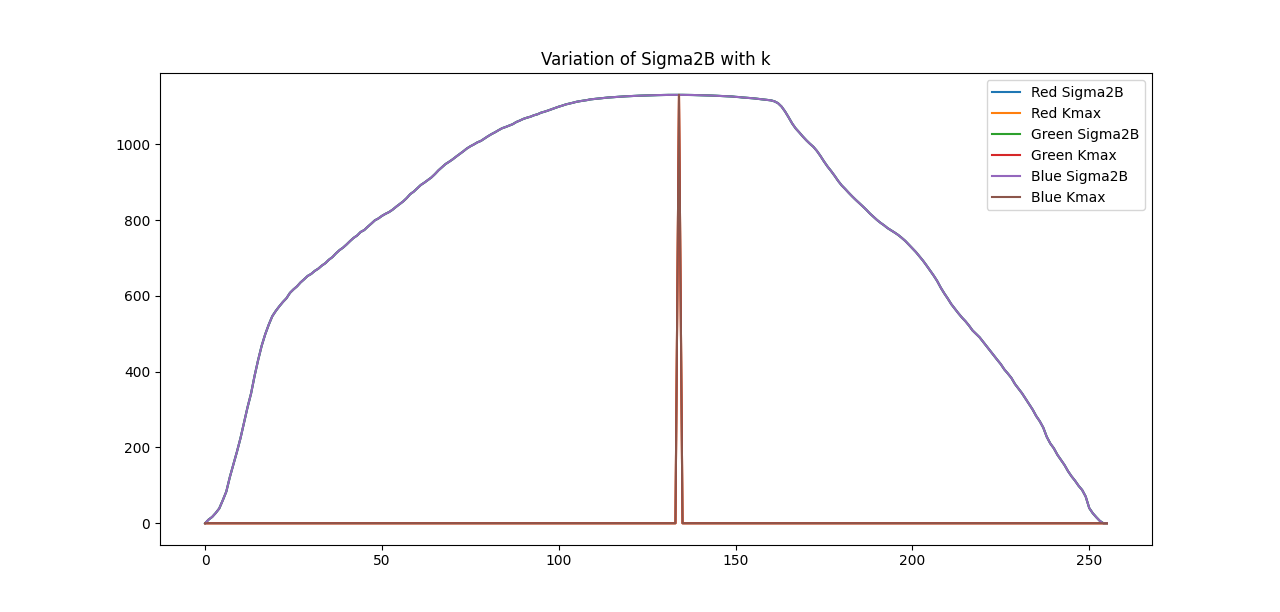
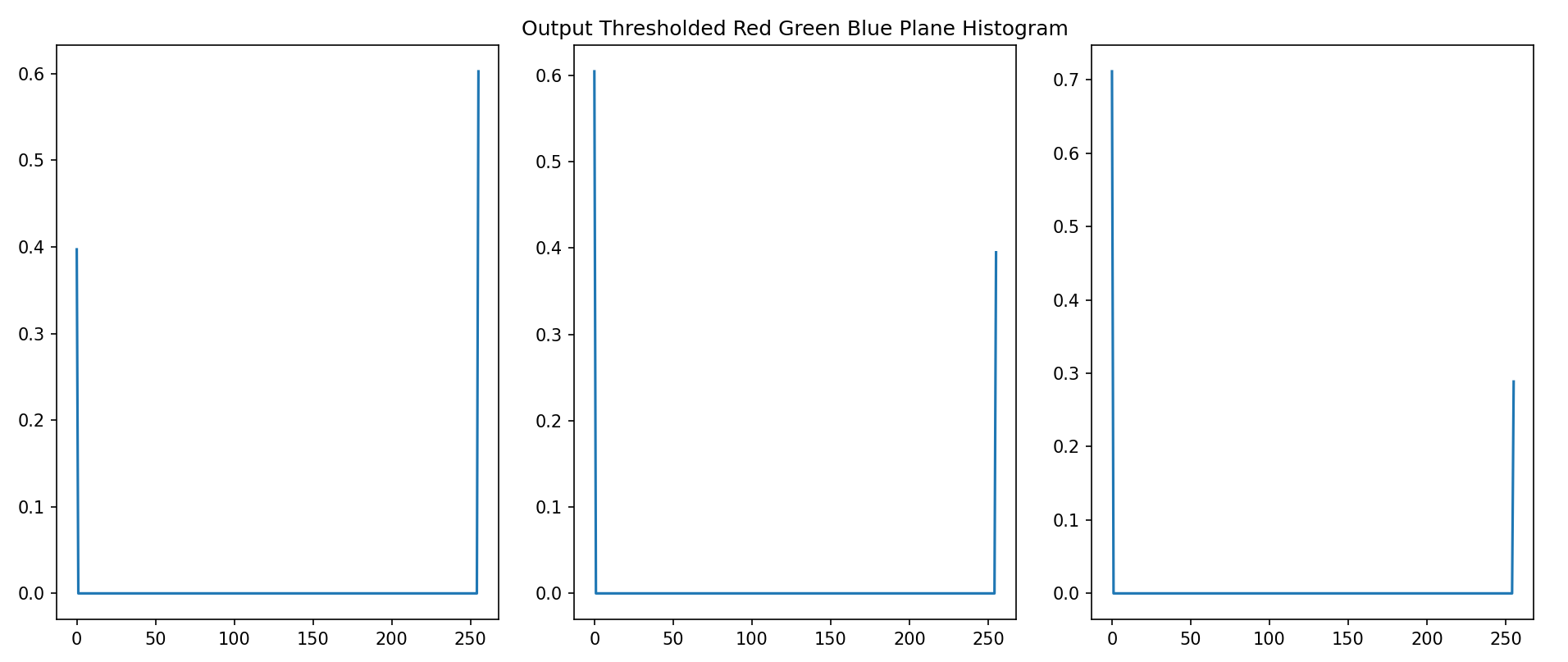
    plt.subplot(1,3,i-8)

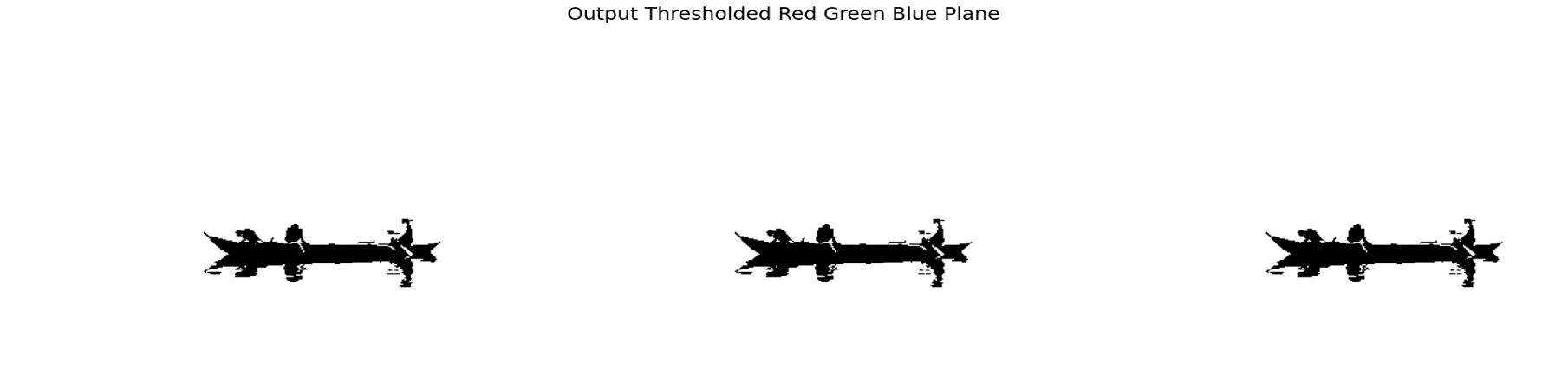
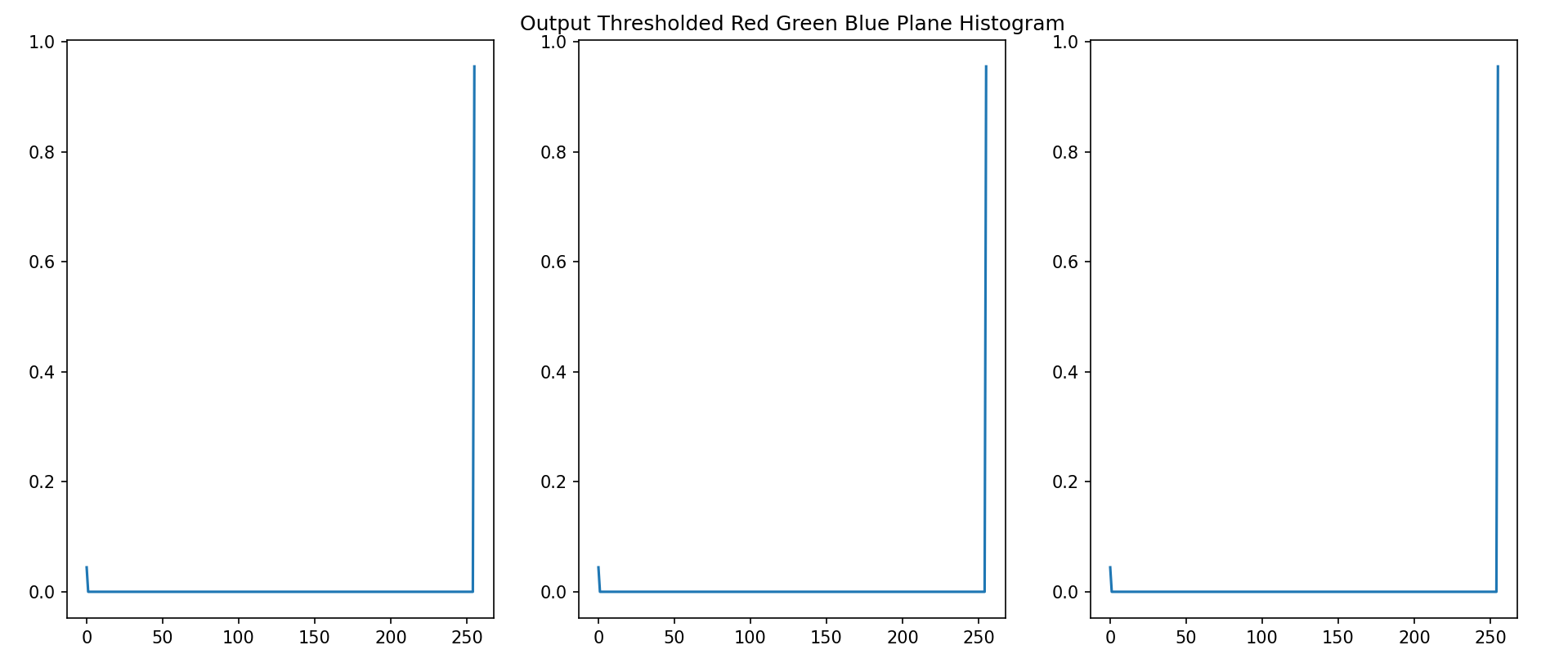
    plt.plot(images[i])

plt.show()

cv2.waitKey(0)

****cv2.destroyAllWindows()

****



**Conclusion:** Otsu’s Method finds the best global threshold for an image by maximizing between-class variance and minimizing within-class variance. This threshold is then used for image segmentation through thresholding, ensuring clear separation between objects and background.